Reactive Databases (with Java +Azure)

https://aka.ms/reactive-java

State Of the Reactive Java Nation

- Jakarta EE & Quarkus
- RXJava
- JDK 9+ →Reactive Steams
- Spring 5
- Spring Boot 2

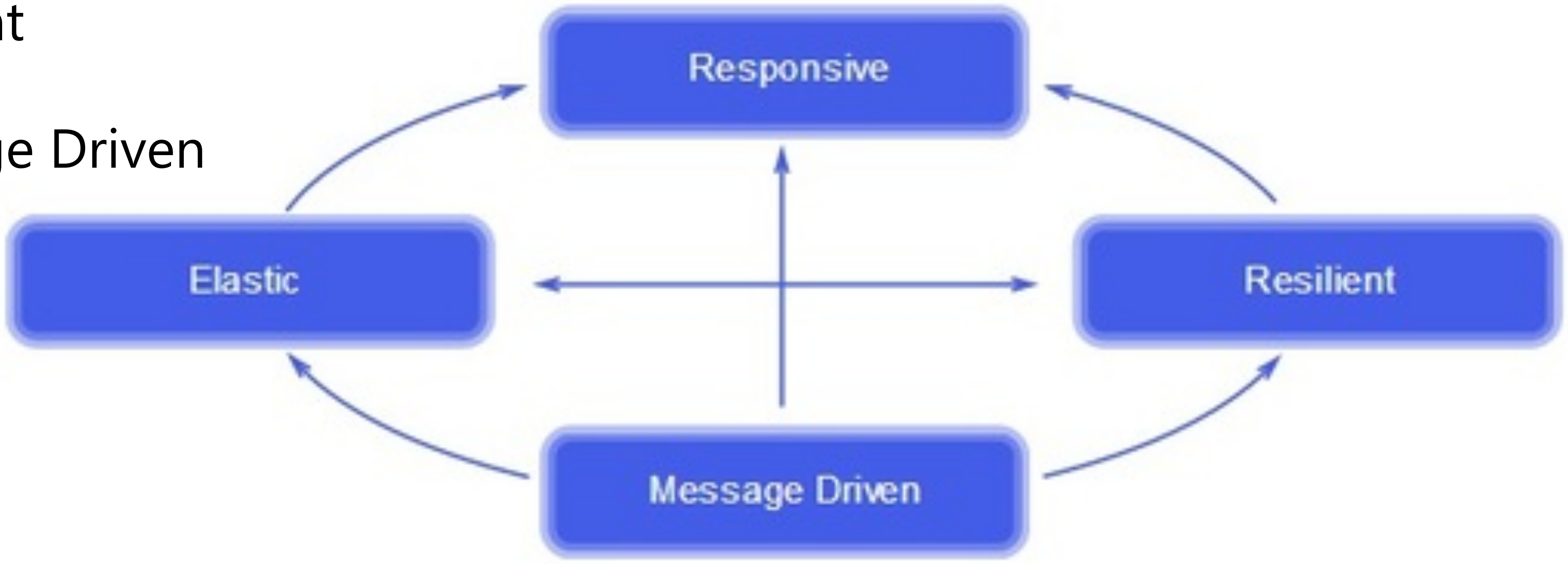
A definition

Reactive Programming is all about non-blocking applications that are asynchronous and event-driven and require a small number of threads to scale

-Spring.io

The Reactive Manifesto

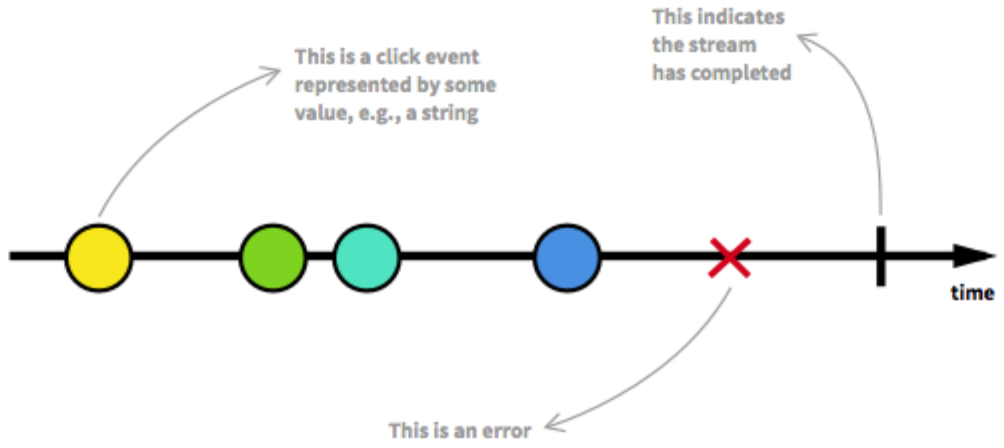
- Responsive
- Resilient
- Elastic
- Message Driven



https://www.reactivemanifesto.org/

What is Reactive Programming?

- ***Observer***
  - Interface to notify an object that the next item in a sequence it is watching it is available
- ***Streams***
  - Controlled exchange of stream data across Applications
- ***Back-pressure***
  - control the flow of a Stream between producer and consumer


This is a click event represented by some value, e.g., a string
This indicates the stream has completed
time
This is an error

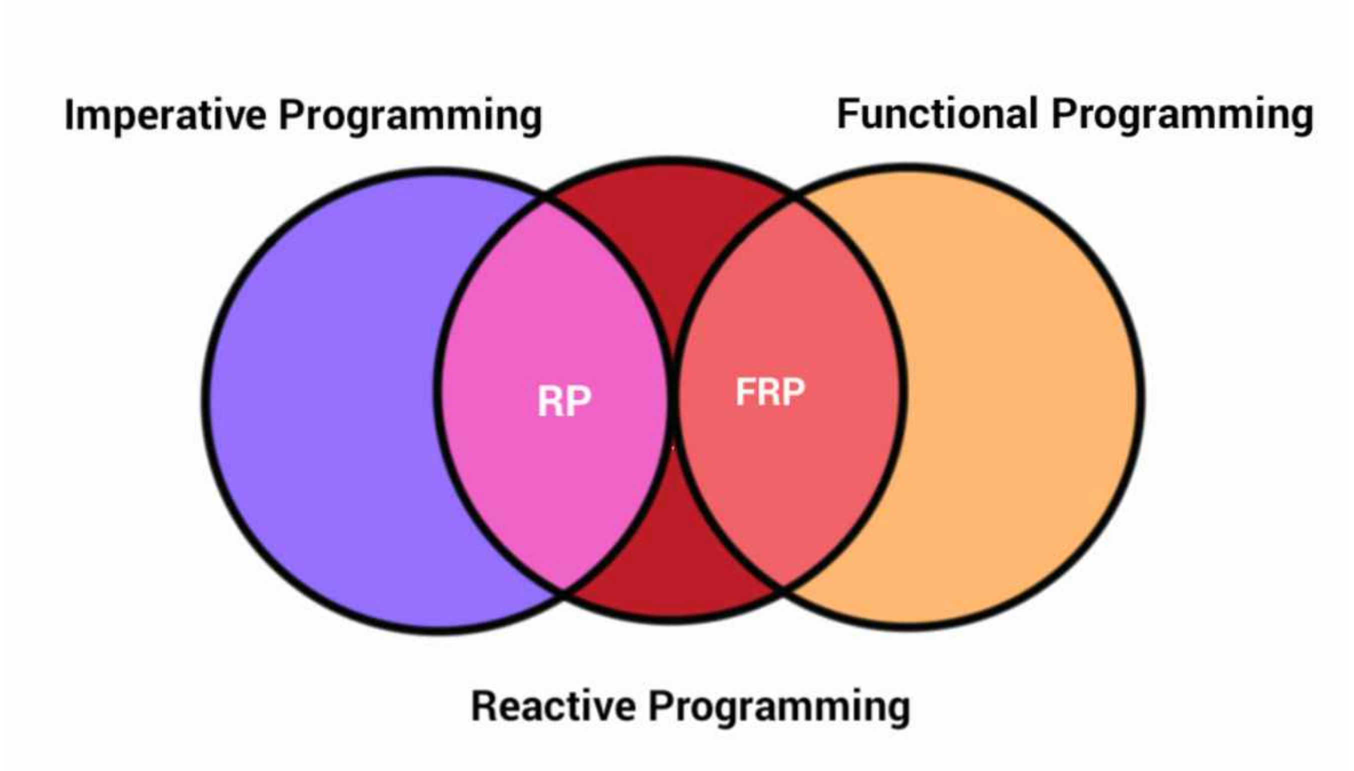
What is Functional Reactive Programming?

Conal Elliot defined FRP back in 1998, in his paper "Functional Reactive Animation”:

- “FRP expressions describe entire evolutions of values over time, representing these evolutions directly as first-class values

What is Functional reactive programming?

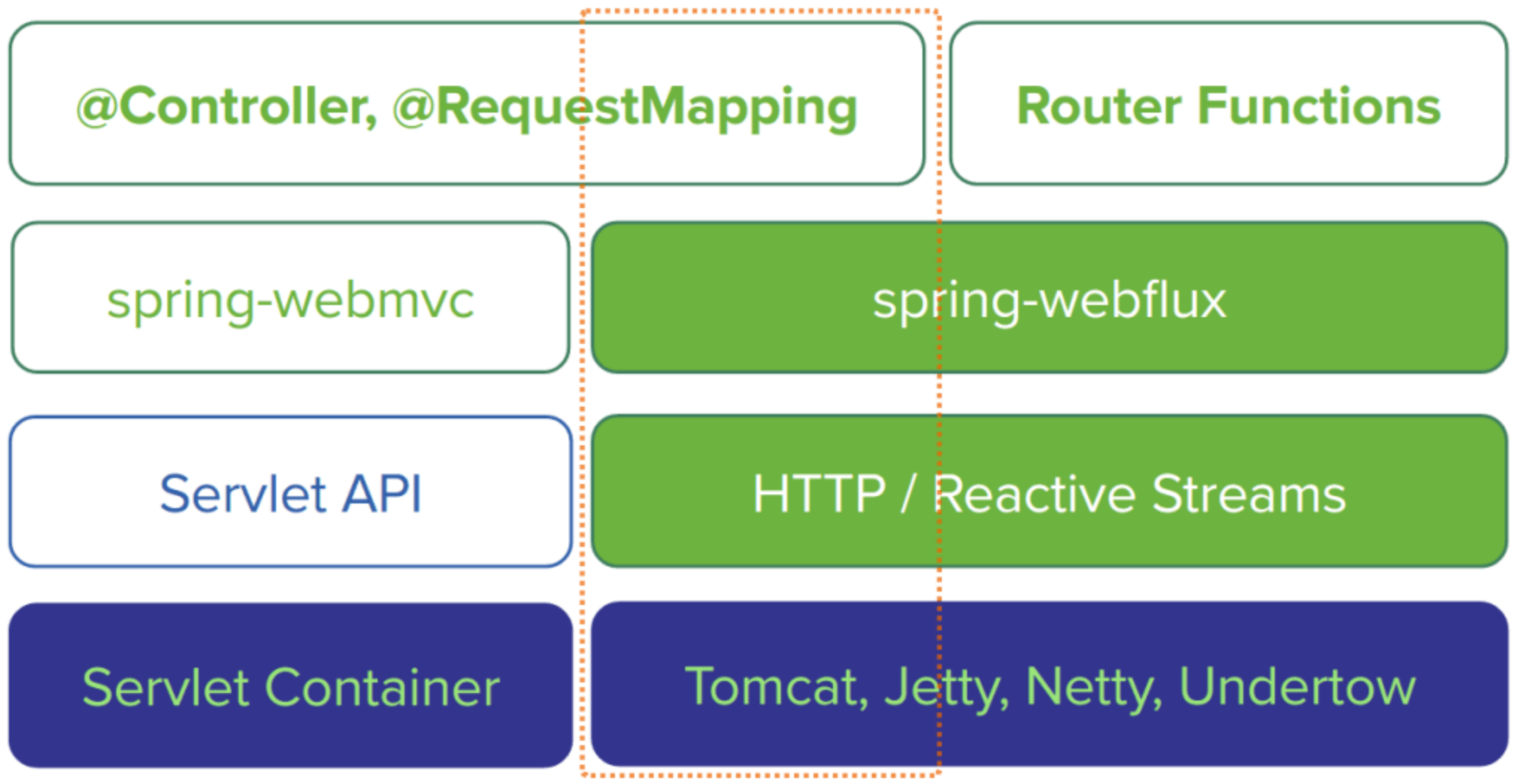
- Compositionality
  - Being able to compose functions
- Immutability
- Guarantees inherently parallelisable



Spring 5

- Java 8+
- Netty
- Webflux
- MVC or Roll-your own http handling

Spring Reactor



Reactive Mongo

```
public interface QuoteMongoReactiveRepository
extends ReactiveCrudRepository<Quote, String> {

}
```

Reactive Rest

```
@GetMapping("/quotes-reactive")
public Flux<Quote> getQuoteFlux() {

    return quoteMongoReactiveRepository.findAll();

}
```

Angular

```
quotes: Quote[] = new Array();
url: string = 'http://localhost:8080/quotes-reactive';

getQuoteStream(page?: number, size?: number): Observable<Array<Quote>> {
  this.quotes = new Array();
  return Observable.create((observer) => {
    let url = this.url;

    let eventSource = new EventSource(url);
    eventSource.onmessage = (event) => {
      console.debug('Received event: ', event);
      let json = JSON.parse(event.data);
      this.quotes.push(new Quote(json['id'], json['book'], json['content']));
      observer.next(this.quotes);
    };
    eventSource.onerror = (error) => observer.error('EventSource error: ' + error);
  });
}
```

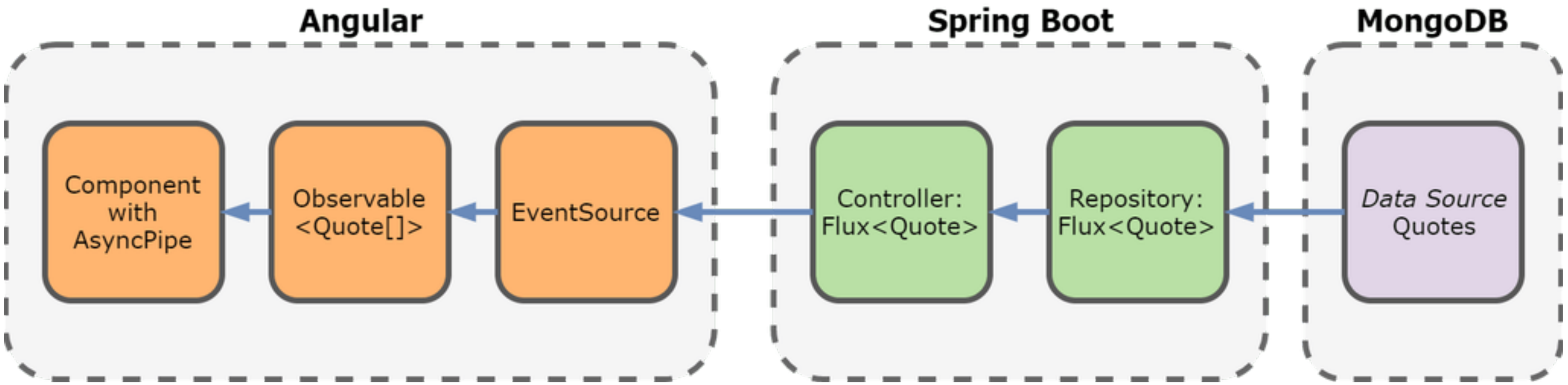
Angular

```
quotes: Quote[] = new Array();
url: string = 'http://localhost:8080/quotes-reactive';

getQuoteStream(page?: number, size?: number): Observable<Array<Quote>> {
  this.quotes = new Array();
  return Observable.create((observer) => {
    let url = this.url;

    let eventSource = new EventSource(url);
     eventSource.onmessage = (event) => {
     console.debug('Received event: ', event);
     let json = JSON.parse(event.data);
      this.quotes.push(new Quote(json['id'], json['book'], json['content']));
     observer.next(this.quotes);
    };
    eventSource.onerror = (error) => observer.error('EventSource error: ' + error);
  });
}
```

Demo Time


Angular
Spring Boot
MongoDB
Component with AsyncPipe
Observable <Quote[]>
EventSource
Controller: Flux<Quote>
Repository: Flux<Quote>
Data Source Quotes

How do you test?

- curl -H "Accept: text/event-stream" http://localhost:8080/quotes-reactive
- Spring WebClient
- Spring WebTestClient
- Postman? Browser?

Pros

- **Responsive**
  - Processing in batches(back-pressure)
- **Resilient** and **Elastic**
  - back-pressure
- **Message Driven**
  - Reactor MicroQueues

Cons

- Immutability?
- Debugging complexity
- Steeeeeeep learning curve
- Do we understand Blocking?
  - “it doesn’t matter if you use a Reactive Web approach in the backend, it won’t be really reactive and non-blocking unless your client is able to handle it as well”

Finally

- Source
  - Java 9+ Flow - https://github.com/reactive-book/java-9-flow
  - Full reactive stack - https://github.com/mechero/full-reactive-stack
- Resources
  - Project Reactor Site - https://projectreactor.io
  - Conal’s original paper https://github.com/conal/talk-2015-essence-and-origins-of-frp

https://aka.ms/reactive-java